Thesis Overview:

Dynamic Tuning of Parallel/Distributed Applications

Anna Morajko
Computer Science Department. Universitat Autònoma de Barcelona. Spain
ania@aows10.uab.es

The main goal of parallel/distributed applications is to solve the considered problem as fast as possible utilizing a certain minimum of the parallel system capacities. In this context, the application performance is one of the most important issues. To satisfy user requirements, applications must reach high performance standards. Therefore, they must not only be systematically tested from the functional point of view to guarantee correctness, but must also be optimized to ensure that there are no performance bottlenecks. The optimization (or tuning) process requires a developer to go through the application performance analysis and the modification of critical application parameters. First, performance measurements must be taken to provide data about the application's behavior. This phase is known as monitoring and collects data related to the execution of the application. Then, the performance analysis of this information is carried out. It finds performance bottlenecks, deduces their causes and determines the actions to be taken to eliminate these bottlenecks. Finally, appropriate changes must be applied to the application code to overcome problems and improve performance. However, all these tasks are somewhat complicated, especially for non-expert users. Current approaches require developers to perform optimizations manually and to have a high degree of experience. Moreover, many applications may have a dynamic behavior. Therefore, it is necessary to provide tools that automatically carry out these tasks. It would be desirable that the performance tuning could be done on the fly by modifying the application according to the particular conditions of the execution.

The thesis addresses the problem of automatic and dynamic tuning of parallel and distributed applications. Our objective is to help developers in the process of improving the application performance. The work presents a whole solution that deals with the issues of automatic and dynamic application improvement. In this approach, an application is monitored, its performance bottlenecks are detected, solutions are given and the application is modified on the fly. All these steps are performed automatically, dynamically and continuously during application execution. This approach exempts developers from performance analysis and difficult intervention to a source code by automatically improving the performance of parallel programs during run-time. The dynamic analysis and introduced modifications permits to adapt the behavior of the application to dynamic variations.

The most useful dynamic tuning is that which can be used to successfully optimize a broad range of different applications. It would be desirable to tune any application even though its source code and application-specific knowledge is not available. However due to incomplete information this kind of tuning is highly challenging and at the same time the most limited. It is not realistic to assume that any modification on any application in any environment can be done on the fly. The key question is: what can be tuned in an "unknown" application?

The answer to this question can be found by investigating how an application is built. Each application consists of several layers: application-specific code, standard and custom libraries, operating system libraries, and hardware. Considering OS and library layers, the tuning process is based on well-known features for them. By investigating particular OS and libraries it is possible to find their potential drawbacks and hence determine problems common to many applications. For each drawback, a tuning procedure can be identified. Optimizing the application code is the most complex and less reusable, due to the lack of application-specific knowledge. Each application implementation can be totally different and there may be no common parts, even though they may provide the same functionality. An application can be tuned if there is knowledge of its internal structure. Therefore, to optimize the application layer, dynamic tuning should be supported in some way with certain information about the application.

Considering the available knowledge, we differentiated two main approaches to tuning: automatic and cooperative. In the automatic approach an application is treated as a black-box, because no application-specific knowledge is provided by the programmer. This approach attempts to tune any application and does not require the developer to prepare it for tuning (the source code does not need to be adapted) and, therefore, it is suitable for tuning such layers as the operating system and libraries. We can find many general tuning procedures common to many applications. For each particular problem, all the necessary information, such as what should be measured, how it should analyzed, and what should be changed and when, can be provided automatically. In the cooperative approach we assume that an application is tunable and adaptable since the developer must

provide application-specific information and prepare an application for the possible changes. Moreover, developers must define an application-specific knowledge that describes what should be measured in the application, what model should be used to evaluate the performance, and finally what can be changed to obtain better performance. The cooperative approach is suitable for the application tuning layer.

To make the presented approaches to tuning (automatic and cooperative), homogeneous and to make optimization on the fly possible and effective we concluded that the application knowledge should be described as the following terms: measure point (a location where the instrumentation must be inserted), performance model (determines an optimal application execution time), tuning point (the code element that may be changed), tuning action (the action to be performed on a tuning point), and synchronization (policy determining when the tuning action can be invoked).

The most important requirements that have to be taken into consideration providing dynamic optimizations are the parallel/distributed application control, the performance on-line analysis, and the run time monitoring and tuning. The principal technique that we could use for dynamic tuning purposes was dynamic instrumentation. By applying this method, it is possible to monitor, analyze and tune a parallel program during run-time. Moreover the application source code is not required. We have devoted big attention to the dynamic instrumentation, in particular to know the library called DynInst that supports this technique.

We wanted to prove experimentally that dynamic tuning is effective and it is very profitable for users to take advantages of a tool that supports the automatic dynamic optimization functionalities. With this objective we have developed an environment called MATE (Monitoring, Analysis and Tuning Environment) that provides dynamic automatic tuning of parallel/distributed applications. MATE performs dynamic tuning in three basic and continuous phases: monitoring, performance analysis and modifications. This environment dynamically and automatically instruments and traces a running application to gather information about the application behavior. The analysis phase searches for bottlenecks, detects their causes and gives solutions on how to overcome them. Finally, the application is dynamically tuned by applying given solution. Moreover, while it is being tuned, the application does not need to be re-compiled, re-linked and restarted. The MATE environment tries to adapt the application to the dynamic behavior. MATE consists of the following main components that cooperate among themselves, controlling and trying to improve the application execution:

- Application Controller (AC) – a daemon-like process that controls the application execution on a given host (management of tasks and machines). It also provides the management of task instrumentation and modification.
- Dynamic monitoring library (DMLib) – a shared library that is dynamically loaded by AC into application tasks to facilitate instrumentation and data collection. The library contains functions that are responsible for registration of events with all required attributes and for delivering them for analysis.
- Analyzer – a process that carries out the application performance analysis, it automatically detects existing performance problems “on the fly” and requests appropriate changes to improve the application performance.

Currently, our environment can be treated as the prototype for complete future implementation. There are still many aspects that remain for considerations and improvements. However, conducting many various practical experiments with MATE we showed that it is possible to dynamically tune applications and obtain benefits. Results of the performed tests were very promising and indicated that our prototype is applicable, effective and can be used for a real improvement of the program performance. Running applications under control of a dynamic tuning system has allowed for adapting their behavior to the existing conditions and improving their functionality. Obviously, all components of our environment cause intrusion and influence into the application execution, but we demonstrated that there are many examples where it is smaller than the profits gained from the performed improvements.

Anna Morajko
ania@aows10.uab.es